PAPER *Special Section on Foundations of Computer Science*

Exact Algorithms for Annotated Edge Dominating Set in Graphs with Degree Bounded by 3

Mingyu XIAO[†] ***and*** **Hiroshi NAGAMOCHI**[††a)], ***Members***

SUMMARY Given a graph $G = (V, E)$ together with a nonnegative integer requirement on vertices $r : V \rightarrow Z_+$, the *annotated edge dominating set* problem is to find a minimum set $M \subseteq E$ such that, each edge in $E - M$ is adjacent to some edge in M, and M contains at least $r(v)$ edges incident on each vertex $v \in V$. The annotated edge dominating set problem is a natural extension of the classical edge dominating set problem, in which the requirement on vertices is zero. The edge dominating set problem is an important graph problem and has been extensively studied. It is well known that the problem is NP-hard, even when the graph is restricted to a planar or bipartite graph with maximum degree 3. In this paper, we show that the annotated edge dominating set problem in graphs with maximum degree 3 can be solved in $O^*(1.2721^n)$ time and polynomial space, where n is the number of vertices in the graph. We also show that there is an $O^*(2.2306^k)$-time polynomial-space algorithm to decide whether a graph with maximum degree 3 has an annotated edge dominating set of size k or not.

key words: *edge dominating sets, exact algorithms, cubic graphs*

1. Introduction

Since we do not know whether $P = NP$ or not, currently the best we can exactly solve NP-complete problems is super-polynomial time algorithms. Although to exactly solve some NP-complete problems we can adopt exhaustive search algorithms, the forbiddingly large running time of the search algorithms makes them impractical even on instances of fairly small size. People wonder whether we can design algorithms that are significantly faster than trivial exhaustive search, though they are still not polynomial-time. Research on exponential-time algorithms for some natural and basic problems, such as independent set [9], [20], coloring [1], exact satisfiability [3] and so on, has a long history. Recently, some other basic graph problems, such as dominating set [8], edge dominating set [19] and feedback set [17], also have drawn much attention in this line of research. Furthermore, to get more understanding of the structural properties of NP-complete problems, people also have interests in exactly solving problems in sparse and low-degree graphs. The independent set problem in degree-3 graphs can be solved in $O^*(1.0854^n)$ time [4], the k-vertex cover problem in degree-3 graphs can be solved in $O^*(1.1616^k)$ time [22], and TSP in cubic graphs can be solved in $O^*(1.251^n)$ time [12]. More fresh results on problems in sparse and low-degree graphs can be found in the literature [5], [10], [18], [21], [26]. To a certain extent, some graph problems in low-degree graphs are the bottleneck of improving the algorithms for the problems in general graphs. Motivated by those, in this paper, we study the classic edge dominating set problem and an extension of it in graphs with degree bounded by 3 and present some improved algorithms for them. Except exact algorithms, we also study *parameterized algorithms* for our problems. In parameterized algorithms, we first pick up a parameter of the problem (the parameter can be the size of the solution, number of the vertices of the input graph, treewidth of the input graph, and so on), and try to design algorithms such that the exponential part of the running time depends on only the parameter (but not the whole input size). We can regard parameterized algorithms as a kind of exact algorithms. Parameterized algorithms for some basic graph problems, including the edge dominating set problem, have been extensively studied recently. For more details about parameterized algorithms, the readers can refer to recent monographs [13]. In this paper, we will take k, the size of the solution to the edge dominating set problem, as the parameter to study the parameterized algorithms.

The edge dominating set problem is a basic problem introduced in Garey and Johnson's work [11] on NP-completeness. Yannakakis and Gavril [28] proved that the edge dominating set problem is NP-hard even in planar or bipartite graphs of maximum degree 3. Randerath and Schiermeyer [15] designed the first nontrivial exact algorithm for the minimum edge dominating set problem, which runs in $O^*(1.4423^m)$ time, where m is the number of edges in the graph. Later Raman *et al.* [14] improved the result to $O^*(1.4423^n)$. Fomin *et al.* [7] claimed an $O^*(1.4082^n)$-time algorithm by considering the treewidth of the graphs. Rooij and Bodlaender [19] got an $O^*(1.3226^n)$-time algorithm by using the 'measure and conquer' method, which was further improved to $O^*(1.3160^n)$ [24]. In terms of parameterized algorithms with parameter k being the size of the solution, there are also a long list of contributions to the upper bound of the running time. Let us quote the $O^*(2.6181^k)$-time algorithm by Fernau [6], the $O^*(2.4181^k)$-time algorithm by Fomin *et al.* [7], the $O^*(2.3819^k)$-time algorithm by Binkele-Raible and Fernau [2], and finally the $O^*(2.3147^k)$-time algorithm by Xiao *et al.* [27]. Most of these algorithms are

Manuscript received March 1, 2012.
Manuscript revised July 28, 2012.

[†]The author is with School of Computer Science and Engineering, University of Electronic Science and Technology of China, China.
[††]The author is with Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: nag@amp.i.kyoto-u.ac.jp

DOI: 10.1587/transinf.E96.D.408

Copyright © 2013 The Institute of Electronics, Information and Communication Engineers

based on the idea of enumerating vertex covers.

In this paper, we study the *annotated edge dominating set problem* in graphs of maximum degree 3 and present fast exact and parameterized algorithms for it. Our exact algorithm is the first effective algorithm that are not based on enumeration of minimal vertex covers and the algorithm is analyzed by measuring the number of degree-3 vertices instead of the number of vertices or edges. We also show that the exact algorithm can be used to derive a fast parameterized algorithm for our parameterized problem. In an initial version of this paper [23], we claimed the results on the edge dominating set problem. In the full version of this paper, we extend the results to the annotated edge dominating set problem.

The rest of the paper is organized as follows: Sect. 2 gives the basic definitions and our notation system. Section 3 gives some reduction rules which will be used as preprocesses to simplify input graphs. Section 4 gives our exact algorithm for the annotated edge dominating set problem in degree-3 graphs. Section 5 designs an algorithm for the parameterized version of this problem. Finally Sect. 6 makes the conclusion.

2. Preliminaries

Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges. A graph is called a *degree-*3 *graph* if any vertex in the graph is of degree ≤ 3. We may use p to denote the number of degree-3 vertices in a degree-3 graph. Let $G[V']$ be the subgraph induced by a subset $V' \subseteq V$, $V(G)$ be the vertex set of graph G, and $E(v)$ be the set of edges incident on vertex v. A subset $S \subseteq E$ *dominates* an edge $e \in E - S$ if it contains an edge adjacent to e, and is called an *edge dominating set* of the graph if it dominates every edge in $E - S$. The *edge dominating set problem* is to find an edge dominating set of minimum size. In the *annotated edge dominating set problem*, a pair (G, r) of a graph G and a nonnegative integer requirement of vertices $r : V \to Z_+$ is given (where Z_+ is the set of nonnegative integers), and we are asked to find a minimum edge dominating set M such that for each vertex $v \in V$, $|M \cap E(v)| \geq r(v)$. An instance $I = (G, r)$ of the annotated edge dominating set problem is simply denoted by G when the requirement of vertices is clear. The original edge dominating set problem is the special case of the annotated edge dominating set problem where the weight of vertices is always zero. For a vertex v, let $N(v)$ denote the neighbor set of v, i.e., the set of vertices adjacent to v, and $N_2(v)$ denote the set of vertices with distance exactly 2 from v. Let $d(v) = |N(v)|$ denote the *degree* of a vertex v, and $N[v] = N(v) \cup \{v\}$ denote the *closed neighbor set* of v.

For an integer $i \geq 0$, a path of length i (or an (i)*-path*) is an ordered list $v_1 v_2 \cdots v_{i+1}$ of distinct i vertices such that there is an edge between v_j and v_{j+1} for all $1 \leq j \leq i$. A path $v_1 v_2 \cdots v_{i+1}$ in G is called an *inner path*, if $v_2, v_3, \ldots, v_i$ are degree-2 vertices of G and v_1 and v_{i+1} are vertices of degree ≥ 2. In addition, if v_1 and v_i are vertices of degree ≥ 3, we also call such an inner path a *maximal inner path*. We say that a maximal inner path $v_1 v_2 \cdots v_{i+1}$ is *incident on* v_1 (or v_{i+1}). For a subset $X \subseteq V(G)$ of vertices, the set $E(X, V(G) - X)$ of edges with one endpoint in X and the other one in $V(G) - X$ is called an i-edge cut of X in G, where $i = |E(X, V(G) - X)|$ is the number of edges in $E(X, V(G) - X)$. Note that we can find all 1-edge cuts and 2-edge cuts in polynomial time. In a degree-3 graph, a 1-edge cut (resp., 2-edge cut) $E(X, V(G) - X)$ of vertex set X is *good* if the induced graph $G[X]$ contains at most 11 (resp., 7) degree-3 vertices.

Annotating a vertex v means that we set $r(v) := \max\{1, r(v)\}$. Note that annotation is the only operation in the paper that increases the requirement of a vertex. It only increases the requirement from 0 to 1, and never increases the requirement to any number greater than 1.

We will use a modified O notation that suppresses all polynomially bounded factors. For two functions f and g, we write $f(n) = O^*(g(n))$ if $f(n) = g(n)n^{O(1)}$.

In this paper, we design algorithms for the annotated edge dominating set problem in graphs with degree bounded by 3. Our algorithms are based on the branch-and-reduce paradigm. We first apply reduction rules to reduce the size of the input. Let I be any instance of our problem and I' be the instance obtained by applying a reduction rule. If a solution to I can be constructed in polynomial time from a solution to I', we say this reduction rule is *correct*. We may find a partial solution directly by using correct reduction rules. When none of our reduction rules can be applied, we will apply some branching rules to search solutions by including some edges into the solution or excluding some edges from the solution. In each branch, we will delete some edges or vertices from the graph and get a new problem instance in a graph with a smaller measure (the measure can be the number of vertices, edges or others). Then a computation process of such an algorithm is represented by a search tree $\mathcal{T}$. At each node of $\mathcal{T}$, the current instance is branched into several small instances. Let $C(p)$ denote the worst size of the search tree $\mathcal{T}$ of an algorithm applied to an instance with measure p. Assume that the algorithm branches on a graph G with measure p at a node in $\mathcal{T}$ into t graphs G_i with measure p_i $(i = 1, 2, \ldots, t)$. Then for the branch, we get the recurrence relation $C(p) \leq \sum_{i=1}^{t} C(p_i)$. Solving the recurrence, we get $C(p) = O(\alpha^p)$, where α is the largest root of the equation $f(x) = 0$ for the function $f(x) = 1 - \sum_{i=1}^{t} x^{-p_i}$. We need to analyze the 'worst' node in the search tree to prove an upper bound of the size of $\mathcal{T}$.

In some steps of our algorithm, we may include some edges into the solution. We show how to handle an instance $I = (G = (V, E), r)$ of the annotated edge dominating set problem with a prescribed edge e that is required to be included in the edge dominating set. We call such an edge *forced*. Let F be a set of forced edges. In fact, such an additional requirement can be eliminated by modifying the instance. Note that just deleting edges in F from G does not give a correct new instance of our problem since the edges $E - F$ adjacent to F are already dominated by F. A new instance $I' = (G', r')$ of the annotated edge dominating set problem is constructed as follows:

Including prescribed edges
For a set $F \subseteq E$ of forces edges, let $E_0 \subseteq E - F$ be the set of edges e such that (i) e is adjacent to an edge in F; (ii) $r(v) \leq |F \cap E(v)|$ for each endpoint v of e. Remove $F \cup E_0$ from G deleting any vertices of degree-0, and set $r'(v) := \max\{0, r(v) - |F \cap E(v)|\}$ for each vertex v to obtain a new instance $I' = (G' = (V, E - F - E_0), r')$.

Lemma 1: Let $I = (G, r)$ be an instance of the annotated edge dominating set problem and $F \subseteq E$ be a prescribed set of edges. Let $I' = (G', r')$ be the new instance obtained by the above rule. A minimum annotated edge dominating set of I that contains F as a subset can be constructed in polynomial time from a solution to I'.

Proof: We call an annotated edge dominating set M of I a *feasible solution* if it satisfies the constraint $F \subseteq M$. Let M' be a minimum annotated edge dominating set of $I' = (G', r')$. We first show that the set $M' \cup F$ is a feasible solution to $I = (G, r)$. By definition of r', $M' \cup F$ satisfies the requirement r in I. Assume that $M' \cup F$ does not dominate some edge $uv \in E - F - M'$. Since $M' \cup F$ satisfies the requirement r, it holds $r(u) = r(v) = 0$, and hence $uv \in E_0$ or $uv \in E - F - E_0$. In the former case, uv is dominated by F, and in the latter, uv is dominated by M', a contradiction. Therefore, $M' \cup F$ is an annotated edge dominating set of I.

We next show that $M' \cup F$ is a minimum feasible solution among all feasible solutions to I. For this, it suffices to prove that there is a minimum feasible solution which contains no edge in E_0. Let M be a minimum feasible solution such that $|M \cap E_0|$ is minimized among all feasible solutions. By definition, F dominates all edges in E_0 and F is contained in M. Let M contain an edge $uv \in E_0$, where $r'(u) = r'(v) = 0$ by definition. Then uv must be adjacent to an edge $e \in E - F - E_0$, since otherwise $M - \{uv\}$ would be a smaller feasible solution to I. In this case, $M \cup \{e\} - \{uv\}$ is an annotated edge dominating set in I, contradicting the minimality of $|M \cap E_0|$. Therefore, I has a minimum feasible solution M such that $M \cap E_0 = \emptyset$, as required. □

In what follows, when we include a set F of edges into a solution, we assume that a new instance $I' = (G', r')$ from $I = (G, r)$ is constructed in the above way unless explicitly stated.

3. Reduction Rules

Reduction rules are frequently used as preprocesses to reduce the input size. We can apply the rules to transform an instance I to an *equivalent* instance I' with smaller measure in polynomial time, i.e., a solution to I can be constructed in polynomial time from a solution to I', and vice versa. In fact, here we only consider one direction: to guarantee a solution to I can be derived from a solution to I'. Since reduction rules are applied to reduce data for instances, we introduce them before presenting our algorithm.

We first observe the correctness of the following two simple rules for vertices v with $r(v) \geq d(v)$.

Rule 1: Overly required vertices
If there is a vertex v such that $r(v) > d(v)$, the instance has no solution.

Rule 2: Fully required vertices
A vertex v is fully required, if $r(v) = d(v)$. If there is a fully required vertex, then we include the set F of all edges incident on every fully required vertex into the solution and reduce the graph according to the removal of F (note that no new fully required vertex will be generated).

Next, we assume that in the graph Rule 1 and Rule 2 have been applied and then for each vertex v in the graph, we have that $r(v) < d(v)$.

To deal with some types of vertices with no requirement, we have the following reduction rule, where we assume that the graph has no component of a path (which component can be solved directly).

Rule 3: Folding some special vertices with no requirement
Let v be a vertex with $r(v) = 0$.
Case 1. *$d(v) = 1$: Remove v from the graph, annotating the unique neighbor u of v;*
Case 2. *$d(v) = 2$ and the two neighbors $u, w \in N(v)$ are adjacent: Remove v from the graph and annotate u and w; and*
Case 3. *$2 \leq d(v) \leq 3$ and $r(u) \geq 1$ for all neighbors $u \in N(v)$: Remove v from the graph without changing $r(u)$ for each $u \in N(v)$.*

Lemma 2: Rule 3 is correct.

Proof: Case 1. It is easy to see that there is a solution S which does not contain edge vu. We annotate the neighbor u of v so that at least one edge incident on u is selected into the solution.

Case 2. If vu or vw is in a solution S, we can simply replace it with uw in S to get another solution, because u and w are not fully required. Then we can assume that $vu, vw \notin S$ and $u, w \in V(S)$. Therefore, we can delete v increasing the requirement of each of u and w to 1 if it is 0 to find such a solution S.

Case 3. Note that each neighbor u of v is not fully required. If vu is in the solution S, then $E(u) - S \neq \emptyset$ (otherwise $S - \{vu\}$ would be a smaller solution) and we can simply replace vu with an edge in $E(u) - S$ to get another solution. Therefore we can assume that $N(v) \cap S = \emptyset$, and we can delete v without losing a solution. □

We introduce a reduction to some kinds of inner paths $abcde$ of length 4. Since b, c and d are degree-2 vertices, the requirements on them can only be 1 or 0. We get the following reduction rules that depend on the requirements on b, c and d (see Fig. 1 for the illustration).

Rule 4: Folding inner paths of length 4
Let $abcde$ be an inner path in a graph.
Case 1. *$r(c) = 0$ and $r(b) + r(d) \leq 1$: Contract $\{a, b, c, d\}$ into the single vertex a deleting any self-loops and multiple*

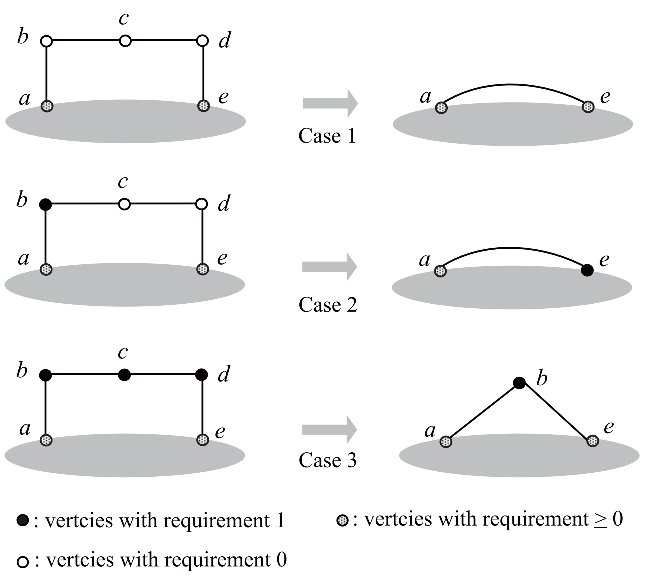
b
c
d
a
e
Case 1
Case 2
Case 3
● : vertcies with requirement 1
◎ : vertcies with requirement ≥ 0
○ : vertcies with requirement 0

Fig. 1 Illustration of reduction Rule 5.

edges without changing $r(a)$;
Case 2. *$r(b) = 1$ (resp., $r(d) = 1$) and the requirements on the other two vertices are 0: Contract $\{b, c, d, e\}$ (resp., $\{a, b, c, d, \}$) into the single vertex e (resp., a) deleting any self-loops and multiple edges, and annotate e (resp., annotate a); and*
Case 3. *$r(b) = r(c) = r(d) = 1$: Contract $\{b, c, d\}$ into the single vertex b with $r(b) = 1$ deleting any self-loops.*

Lemma 3: Rule 4 is correct.

Proof: Let G be the original graph and G' the graph after applying the reduction rule on inner path $P = abcde$. We prove that G has an annotated edge dominating set S of size k if and only if G' has an annotated edge dominating set S' of size $k - 1$ and the two solutions can be derived from each other easily. Let S denote a solution in each of Cases 1-3 such that $|S \cap E(P)|$ is minimized.

First, we consider Cases 1 and 2 ($r(b) = 1$ is assumed in Case 2 without loss of generality). (i) Only if part: Since $r(c) = 0$ and $r(b) + r(d) \leq 1$, we see that $S \cap E(P) = \{bc\}$ or $\{cd\}$ when $|S \cap E(P)| = 1$; and $S \cap E(P) = \{ab, de\}$ when $|S \cap E(P)| \geq 2$ (since otherwise we can replace $S \cap E(P)$ with $\{ab, de\}$ to obtain another solution). Note that $e \in V(S)$ always hold when $bc \in S$. Then in the former, $S' = S - \{bc\}$ is an annotated edge dominating set of size $k-1$ in G', while $S' = (S - \{ab, de\}) \cup \{ae\}$ will be a solution of size $k - 1$ to G' in the latter.

(ii) If part: For an annotated edge dominating set S' of size $k - 1$ in G', we consider three cases: $ae \in S'$; $ae \notin S'$ and $e \in V(S')$; and $\{a, e\} \cap V(S') = \{a\}$: If $ae \in S'$, then $S = (S' - \{ae\}) \cup \{ab, de\}$ is an annotated edge dominating set of size k for G. $ae \notin S'$ and $e \in V(S')$ If $ae \notin S'$ and $e \in V(S')$, then $S = S' \cup \{bc\}$ is an edge dominating set of size k for G. For the third case, $S = S' \cup \{cd\}$ is an edge dominating set of size k for G in Case 1, whereas $S = S' \cup \{bc\}$ is an edge dominating set of size k for G in Case 2 (note that $r(e) \geq 1$ in Case 2).

Next, we consider Case 3. (i) Only if part: For an annotated edge dominating set S of size k in G, consider two cases: $ab \in S$ and $bc \in S$. If $ab \in S$, we can assume that $cd \in S$. Then $S' = S - \{cd\}$ is an annotated edge dominating set of size $k - 1$ for G'. Next if $bc \in S$, we can assume that $de \in S$. In this case, $S' = (S - \{bc, de\}) \cup \{be\}$ is an annotated edge dominating set of size $k - 1$ for G'.

(ii) If part: For an annotated edge dominating set S' of size $k-1$ in G', we consider two cases: $ab \in S'$; and $be \in S'$. If $ab \in S'$, then $S = S' \cup \{cd\}$ is an edge dominating set of size k for G. If $be \in S'$, then $S = (S' - \{be\}) \cup \{bc, de\}$ is an edge dominating set of size k for G. □

Note that the case of $r(b) = r(d) = 1$ and $r(c) = 0$ cannot occur for an inner path $abcde$ by Rule 3 being applied to c. We also note that Rule 4 cannot be used to reduce all inner paths of length 4, because these cases do not cover all cases. As shown in the next lemma, we can use Rule 4 to reduce all inner paths of length ≥ 6.

Lemma 4: In a graph after applying Reduction Rules 1 to 4, there is no inner path of length ≥ 6 and any inner path of length 4 or 5 satisfies one of the follows:
(i) inner path $abcde$ of length 4 such that $r(a) = r(b) = 0$ and $r(c) = r(d) = 1$ (or $r(d) = r(e) = 0$ and $r(b) = r(c) = 1$);
(ii) inner path $abcde$ of length 4 such that $r(c) = 1$ and $r(a) = r(b) = r(d) = r(e) = 0$; and
(iii) inner path $abcdef$ of length 5 such that $r(c) = r(d) = 1$ and $r(a) = r(b) = r(e) = r(f) = 0$.

Proof: It is easy to see that any inner path of length 4 other than (i) and (ii) can be reduced by Reduction Rule 3 or 4. Let $abcdef$ be an inner path of length 5. Since its inner paths $abcde$ and $bcdef$ of length 4 satisfy (i) or (ii), only the case of (iii) is a possible configuration. If there is an inner path $u_1u_2 \cdots u_{i+1}$ of length $i \geq 6$, then its inner paths $u_1u_2 \cdots u_5$ and $u_2u_3 \cdots u_6$ of length 5 need to satisfy (iii), which is impossible. □

Next, we show that when the graph has a 1-edge cut, we can deal with the graph in an easy way. We observe the following properties of 1-edge cuts.

Lemma 5: Let aa' be a 1-edge cut of a set X in instance $I = (G, r)$, where $a \in X$ and $a' \in V(G) - X$. Assume that $r(a) = 2$. Let $I_2 = (G[X], r)$ be the annotated edge dominating set instance on the induced graph $G[X]$, and $I_1 = (G[X], r')$ be the annotated edge dominating set instance on the induced graph $G[X]$ with $r'(a) = 1$ and $r'(v) = r(v)$ for $v \in X - \{a\}$. Let F_1 and F_2 denote solutions to I_1 and I_2, respectively. If $|F_1| = |F_2|$ (resp., $|F_1| < |F_2|$), then there is a solution M to I such that $F_2 \subseteq M$ (resp., $F_1 \subseteq M$).

Proof: Let $E_X = E(G[X])$, and M be a solution to I. Clearly it holds $|M \cap E_X| \geq |F_1|$, since $d(a) = 3$ and $r(a) = 2$ (we always assume that Rules 1-2 can not be applied any more). Hence if $|F_1| = |F_2|$, then we can replace $M \cap E_X$ with F_2 in M to obtain another solution to I. Let $|F_1| < |F_2|$. Then we can replace $M \cap (E_X \cup \{aa'\})$ with $F_1 \cup \{aa'\}$ in M to obtain another solution to I (even if $aa' \notin M$). □

Lemma 6: Let aa' be a 1-edge cut of a set X in instance $I = (G, r)$, where $a \in X$ and $a' \in V(G) - X$. Assume that $r(a) = 1$. Let $X_0 = X - \{a\}$, $I_1 = (G[X], r)$ be the annotated edge dominating set instance on the induced graph $G[X]$, and $I_0 = (G[X_0], r)$ be the annotated edge dominating set instance on the induced graph $G[X_0]$. Let F_0 and F_1 denote solutions to I_0 and I_1, respectively. If $|F_0| = |F_1|$ (resp., $|F_0| < |F_1|$), then there is a solution M to I such that $F_1 \subseteq M$ (resp., $F_0 \subseteq M$).

Proof: Let $E_X = E(G[X])$, and M be a solution to I. Clearly it holds $|M \cap E_X| \geq |F_0|$, since $X_0 \subseteq X$. Hence if $|F_0| = |F_1|$, then we can replace $M \cap E_X$ with F_1 in M to obtain another solution to I. Let $|F_0| < |F_1|$. Then we can replace $M \cap (E_X \cup \{aa'\})$ with $F_0 \cup \{aa'\}$ in M to obtain another solution to I (even if $aa' \notin M$). □

Lemma 7: Let aa' be a 1-edge cut of a set X in instance $I = (G, r)$, where $a \in X$ and $a' \in V(G) - X$. Assume that $r(a) = 0$. Let $X_0 = X - \{a\}$, $I_0 = (G[X_0], r)$ be the annotated edge dominating set instance on the induced graph $G[X_0]$ and $I'_0 = (G[X_0], r')$ be the annotated edge dominating set instance obtained from I_0 by annotating all vertices in $X_0 \cap N(a)$. Let $I_1 = (G[X], r_1)$ be the annotated edge dominating set instance on the induced graph $G[X]$ obtained by annotating a in the instance $(G[X], r)$. Let F_0, F'_0 and F_1 denote solutions to I_0, I'_0 and I_1, respectively. If $|F_0| = |F_1|$, then there is a solution M to I such that $F_1 \subseteq M$. Otherwise ($|F_0| < |F_1|$) if $|F'_0| > |F_0|$ (resp., $|F'_0| = |F_0|$), then there is a solution M to I such that $F_0 \subseteq M$ (resp., $F'_0 \subseteq M$).

Proof: Let $E_X = E(G[X])$, and M be a solution to I. Clearly it holds $|M \cap E_X| \geq |F_0|$, since $X_0 \subseteq X$. Hence if $|F_0| = |F_1|$, then we can replace $M \cap E_X$ with F_1 to obtain another solution to I. Let $|F_1| \geq |F_0| + 1$. If M contains $|F_0| + 1$ (or more) edges from E_X, then we can replace these $M \cap E_X$ with $F_0 \cup \{aa'\}$ to get another solution. Assume that M contains exactly $|F_0|$ edges from E_X. If M contains edge aa', then we can replace $M \cap E_X$ with F_0 to get another solution. Suppose that $aa' \notin M$. Then we see that M contains no edge between a and X_0, since otherwise M would contain $|F_1| > |F_0|$ edges from E_X. Then $M \cap E_X$ does not contain any edge between a and X_0, but dominates all of these edges. This implies that $|F'_0| = |M \cap E_X| = |F_0|$. Hence we can replace $M \cap E_X$ with F'_0 to get another solution. □

The above three lemmas tell us that when the graph has a 1-edge cut we can use a divide-and-conquer method to solve the problem. In our algorithms, we only use this method to reduce good 1-edge cuts, where the induced graph $G[X]$ contains at most 11 degree-3 vertices. After applying Reduction Rule 4, any component G' containing only a constant number of degree-3 vertices can have a constant number of vertices (since after applying Reduction Rule 4, all maximal inner paths are of length at most 5 by Lemma 4). Therefore, the instances I_0, I'_0, I_1 and I_2 in the above three lemmas can be solved in polynomial time. We can reduce the graph to a small instance in polynomial time according to the above three lemmas, if the graph contains a good 1-edge cut. Note that a component with a vertex of maximum degree 3 contains a good 1-edge cut if there is a maximal inner path $u_1u_2 \ldots u_{i+1}$ of length $i \geq 2$ (but $i \leq 5$) such that u_1 is equal to u_{i+1}.

Rule 5: Dealing with good 1-edge cuts
If there is a good 1-edge cut, reduce the graph according to Lemmas 5 to 7.

Note that after applying Rule 5 on a good 1-edge cut of X, we further apply reduction rules to reduce any resulting degree-1 vertices, which removes all vertices in X completely from the graph. For convenience, we will call a graph a *reduced graph* if none of Rules 1 to 5 can be applied.

4. The Exact Algorithm

To effectively analyze our algorithm, we may require that when applying some branching rules the graph does not contain any good i-edge cuts with $i \leq 2$. Rule 5 can reduce good 1-edge cuts. For good 2-edge cuts, a similar reduction rule could become very complicated. Then we use an effective branching rule to branch on good 2-edge cuts before applying other branching rules, if this kind of local structures exists.

The main steps of our recursive algorithm are presented in Fig. 2. Explanation of each branching operation in Steps 3-8 will be given in the subsequent subsections. During an execution of our algorithm, a subset E' of edges which are decided to be included in an annotated edge dominating set is maintained and we will use S^* to denote a minimum annotated edge dominating set in the input graph that

```
Input: A degree-3 graph and a requirement $r$.
Output: A minimum annotated edge dominating set.

 1. If {The current graph is not a reduced graph},
    apply Reduction Rules 1-5 until obtaining a reduced graph.
 2. If{There is some component of at most 12
    degree-3 vertices}, find solutions to this kind of
    components directly.
 3. Elseif {There is a vertex with requirement 2},
    pick such a vertex and branch on an edge incident
    on it.
 4. Elseif{There is a good 2-edge cut}, branch on
    it.
 5. Elseif{There is a maximal inner path of length
    3}, branch on it.
 6. Elseif{There is a maximal inner path of length
    $\geq 4$}, branch on it.
 7. Elseif{There is a maximal inner path of length
    2}, branch on it.
 8. Elseif{The graph is a 3-regular graph}, branch
    on it.
 9. Compute a solution and return it.
```

Fig. 2 Algorithm *PEDS*(G, r).

contains all the edges in E' if such kind of annotated edge dominating sets exist.

In Step 2, any component with at most 12 degree-3 vertices has a constant number of vertices, since the length of any maximal inner path is at most 5 by Lemma 4.

To analyze the running time of our algorithm, we derive an upper bound on the size of the search tree for computations of our algorithm. Traditionally, we may use the number of vertices or edges in the graph to measure the size. Some recent references [16], [21] used the number of degree-3 vertices as the measure to analyze the running time for the independent set and vertex cover problems in degree-3 graphs. We will also use this technique to analyze our algorithm and get the improved running time bound. Recall that we use p to denote the number of degree-3 vertices in the graph. Note that the annotated edge dominating problem with $p = 0$ can be solved in polynomial time easily. After applying Reduction Rules 1-5 in Step 1, the number of degree-3 vertices in the graph will not increase. We will guarantee that the number of degree-3 vertices will decrease in each of Step 3 to 8. Then the algorithm always successfully halts.

To show the correctness of the algorithm and derive a small bound on the size of the search tree generated by the algorithm, we need a careful analysis for Step 3 to 8. Each of these steps branches on the current instance to generate several new subinstances such that a solution to the current instance can be obtained from solutions to the subinstances. Each subinstance is obtained by deleting a degree-3 vertex or an inner path from the current instance. We examine how many degree-3 vertices will disappear in each subinstance to build an effective recurrence. The following three lemmas are important for counting the number of degree-3 vertices in our analysis.

Lemma 8: Let G be a connected graph with maximum degree 3, and V_i be the set of degree-i vertices in G. Assume that $|V_3| \geq |V_1|$. Then after iteratively applying Reduction Rules 2-3 to G until no degree-1 vertex is left, the number of degree-3 vertices decreases by at least $|V_1|$.

Proof: Let G' be the resulting graph after iteratively applying Reduction Rules 2-3 until there is no degree-1 vertex. Let $V' = V(G')$ and $V'' = V - V'$ be the set of deleted vertices, where it holds $V_1 \subseteq V''$ since no degree-1 vertex is left. Assume $V' \neq \emptyset$ (otherwise we have $V_3 \subseteq V''$ and the lemma obviously holds because $|V_3| \geq |V_1|$). Let Y be the set of vertices in V' which are adjacent to V'' in G. Since every vertex in G' is of degree 2 or 3, no two vertices in V'' are adjacent to the same vertex in Y in G. Hence every vertex in Y is of degree 3 in G but of degree 2 in G'. It suffices to prove that G has at least $|V_1| - |Y|$ degree-3 vertices in V''. To prove that, we first construct a new graph G^* from G by deleting all edges in $G[V']$ and vertices in $V' - Y$. Then the vertices in Y are all degree-1 vertices in G^* and the number c of components in G^* is at most $|Y|$, since G is connected. Note that any tree with bounded degree 3 contains at least $L - 2$ degree-3 vertices if it has L degree-1 vertices. Hence a maximal spanning forest T of G^* contains at least $(|V_1| + |Y|) - 2 - 2(|Y| - 1) = |V_1| - |Y|$ degree-3 vertices, since $|V_1| + |Y|$ is the number of degree-1 vertices in G^* and two trees can be joined into a tree introducing at most two new degree-3 vertices. Therefore, after removing $V - V'$ by Reduction Rules 2-3, the number of degree-3 vertices decreases by at least $|Y| + (|V_1| - |Y|) = |V_1|$. □

Lemma 9: Let G be a connected graph with maximum degree 3 and minimum degree 2 which has no good 1-edge cut. Assume that G contains at least 4 degree-3 vertices. Then after deleting a degree-3 vertex v from G and iteratively applying reduction rules, the resulting reduced graph has at least 4 less degree-3 vertices than G has.

Proof: Let G' be the graph obtained from G by deleting v from G. If G' has no degree-1 vertex, then all neighbors of v are of degree-3 and this implies the lemma. Let G' have a component H containing exactly $j \in \{1, 2, 3\}$ degree-1 vertices. For $j = 3$, graph H is a connected graph G' with at least $4 - 1 = 3$ degree-3 vertices by assumption on G. For $j = 1$, graph H must contain at least one degree-3 vertex in it. For $j = 2$, if H contain at most one degree-3 vertex in it, then this means that H is a path and G would have a good 1-cut of $X = V(H) \cup \{v\}$ in G, a contradiction. In any case H contains at least j degree-3 vertices, and Lemma 9 follows from Lemma 8. □

Lemma 10: Let G be a connected graph with maximum degree 3 and minimum degree 2 which has no good 1-edge cut and has at least 12 degree-3 vertices. Assume that G has a set $Z \subseteq V(G)$ containing $q \leq 8$ degree-3 vertices such that the induced graph $G[Z]$ is a connected graph and any edge between Z and $V(G) - Z$ is incident on two different vertices in Z. Let $t \in \{2, 3, 4\}$ be the number of edges between Z and $V(G) - Z$, and assume that G has no good 2-edge cut when $t = 4$. If we remove the vertices in Z from G and continue to apply reduction rules to the resulting graph until a reduced graph G^* is obtained, then G^* has at least $q + t$ less degree-3 vertices than G has.

Proof: We can reduce q degree-3 vertices from Z. Let $u_1, u_2 \in Z$ be the two vertices such that each edge between Z and $V(G) - Z$ is incident on either u_1 or u_2. Let G' be the resulting graph after deleting Z from G. Note that if u_1 and u_2 are adjacent on a same vertex w not in Z, then $d(w) = 3$ in G (otherwise G would have a good 1-edge cut of $Z \cup \{w\}$ or a good 2-edge cut when $t = 4$) and we regard that one of u_1w and u_2w reduces the degree-3 vertex u, and the other corresponds to the resulting degree-1 vertex w in G'. By Lemma 8, it suffices to show that, for each component H of G', if H contains j degree-1 vertices then it contains at least j vertices whose degree is 3 in G'.

Case 1. H contains four degree-1 vertices: Then G' has only one component and the component $H = G'$ contains at least $12 - t \geq 8$ degree-3 vertices (G contains at least 12 degree-3 vertices).

Case 2. H contains exactly three degree-1 vertices: If

G' has only one component and the component $H = G'$ contains at least $12 - t \geq 8$ degree-3 vertices. Otherwise G' has exactly two components H and H', since there are at most 4 edges between Z and $V(G) - Z$. If H contains at most 2 degree-3 vertices, then $G[Z \cup V(H)]$ contains at most $8+1+2 = 11$ degree-3 vertices (at most one degree-1 vertex in H can be a degree-3 vertex in $G[Z \cup V(H)]$) and the edge between H' and Z would a good 1-cut of $Z \cup V(H)$.

Case 3. H contains exactly two degree-1 vertices, say w_1 and w_2: Assume that H contains at most one degree-3 vertex to derive a contradiction. Then the set of two edges between $\{w_1, w_2\}$ and $V(H) - \{w_1, w_2\}$ is a good 2-edge cut of $V(H) - \{w_1, w_2\}$ in G, and $t \leq 3$ by the assumption of G. Since G has at least 12 degree-3 vertices, G' has another component H'. By $t \leq 3$, there is only one edge between H' and Z, and this edge is a good 1-edge cut of $Z \cup V(H)$ since $Z \cup V(H)$ contains at most $q + 1 \leq 8 + 1 = 9$ degree-3 vertices, a contradiction.

Case 4. H contains only one degree-1 vertex: Then H must contain a vertex of degree greater than 2. □

Lemma 11: Let G be reduced graph after Step 4 (having no good 1- or 2-edge cut) and $P_i = v_1v_2 \cdots v_{i+1}$ an inner path of length $i \geq 1$ in it. Let G' be the component containing the inner path P_i and $q \in \{0, 1, 2\}$ be the number of degree-3 vertices in $\{v_1, v_{i+1}\}$. If we remove the vertices $Z = \{v_1, \ldots, v_{i+1}\}$ in P_i from G and continue to apply reduction rules to the resulting graph until a reduced graph G^* is obtained, then G^* has at least $2 + 2q$ less degree-3 vertices than G has.

Proof: When $i \geq 2$, vertices v_1 and v_{i+1} are not adjacent to each other, otherwise G would have a good 1- or 2-edge cut of Z. Then there are $t = 2+q$ edges between Z and $V(G)-Z$. The lemma follows from Lemma 10. □

Now we are ready to describe the branching operations in Fig. 2 and their analysis. We assume that the reduction rules are applied automatically after execution of any branching operation.

4.1 Branching on Requirement-2 Vertices (Step 3)

In this step, the graph G is a reduced graph. Then any vertex with requirement 2 is a degree-3 vertex. Let a be a vertex with requirement 2 in Step 3. Let G' be the component of G containing a and $N(a) = \{b, c, d\}$ be the set of neighbors of a. Our algorithm will branch on an edge incident on a, say ab, into two branches by excluding ab from the solution or including it into the solution.

In the first branch which excludes ab from the solution, the two edges ac and ad must be included into the solution due to $r(a) = 2$. Then we will delete degree-3 vertex a from the graph according to our reduction rules. By Lemma 9, this reduces at least 4 degree-3 vertices in this branch.

In the second branch which includes ab into the solution, we will delete ab from the graph and update the requirement on a and b. If b is also a degree-3 vertex, we can reduce two degree-3 vertices a and b. Otherwise b is a degree-2 vertex, we will reduce only one degree-3 vertex a and get a degree-1 vertex after deleting ab. By Lemma 8, totally we still can reduce at least two degree-3 vertices in this branch after applying reduction rules.

Let $C(p)$ be the worst size of the search tree when the graph has at most p degree-3 vertices. Then we get the following recurrence for the above branching operation:

$$C(p) \leq C(p - 4) + C(p - 2), \tag{1}$$

which solves to $C(p) = O(1.2721^p)$.

After Step 3, the graph will never contain a vertex with requirement ≥ 2, since the requirement of each vertex will not increase to 2 in our algorithm. In Steps 4-8, if there is a branch of including a single edge $e = uv$ into the solution, then we see that such a solution does not need to contain any edge adjacent to the edge e, and we remove not only edge e but also the both endpoints u and v. Most of the analysis below is based on this property.

4.2 Branching on Good 2-Edge Cuts (Step 4)

Let $E' = \{x_1y_1, x_2y_2\}$ be a good 2-edge cut of X in graph G, where $x_1, x_2 \in X$ are degree-3 vertices and $y_1, y_2 \in V(G)-X$. Let $P = z_1z_2 \cdots y_1x_1$ denote the maximal inner path containing edge x_1y_1, where $z_1 = y_1$ and $z_2 = x_1$ (when P is of length 1) or $z_2 = y_1$ (when P is of length 2). Note that $z_1 \neq x_2$ (otherwise X and $V(P)$ would induce a component with at most 7 degree-3 vertices). Recall that S^* stands for an optimal solution that contains the current solution. We branch on the graph into three branches according to the following cases: $z_1 \notin V(S^*)$; $z_1z_2 \in S^*$; and $z_1 \in V(S^*)$ but $z_1z_2 \notin S^*$. We claim that each of the first and second branches reduces the number p of degree-3 vertices by at least 6 while the third reduces p by at least 4.

For the first branch ($z_1 \notin V(S^*)$), we delete z_1 from the graph, apply Reduction Rule 4 to reduce good 1-edge cut x_2y_2 in the remaining graph, and apply other reduction rules to further reduce the graph. Note that $Z = V(P) \cup X$ will be removed from the graph. Let q be the number of degree-3 vertices in Z, where $3 \leq q \leq 8$, since $z_1, x_1, x_2 \in Z$ and E' is a good 2-edge cut. Now there are $t = 3$ edges between Z and $V(G) - Z$. By Lemma 10, we can reduce p by at least $q + t \geq 6$ in this branch.

For the second branch ($z_1z_2 \in S^*$), we will delete the z_1z_2 together with all edges adjacent to it from the graph, apply Reduction Rule 4 to reduce good 1-edge cut x_2y_2 in the remaining graph, and apply other reduction rules to further reduce the graph. This also removes $Z = V(P) \cup X$ from the graph. According to the analysis for the first branch, we know that we can also reduce p by at least 6 in this branch.

For the third branch ($z_1 \in V(S^*)$ but $z_1z_2 \notin S^*$), we annotate z_1, delete edge z_1z_2, and apply Reduction Rule 4 to reduce good 1-edge cut x_2y_2 and other reduction rules. In this branch, we will delete $Z = V(P) \cup X - \{z_1\}$ from the graph. Since Z contains $q \geq 2$ degree-3 vertices and there are $t = 2$ edges z_1z_2 and x_2y_2 between Z and $V(G) - Z$. By

Lemma 10, in the third branch, we can reduce p by at least 4. Therefore we can branch on a good 2-edge cut with the following recurrence

$$C(p) \leq 2C(p-6) + C(p-4), \tag{2}$$

which solves to $C(p) = O(1.2335^p)$.

After Step 4, the graph has no good 1- or 2-edge cuts and Lemma 11 can be used.

4.3 Branching on Maximal Inner Paths of Length 3 (Step 5)

Assume that the algorithm selects a maximal inner path $abcd$ of length 3 in Step 5. We distinguish three cases according to the requirements of b and c.

If $r(b) = r(c) = 0$, we branch into three branches by including each of ab, bc and cd into the solution. Note that in the branch where bc is included into the solution, we can assume that $a, d \notin V(S^*)$ (for this case $r(a) = r(d) = 0$), because if $a \in V(S^*)$ (or $d \in V(S^*)$) we can replace bc with cd (or replace bc with ab) in S^* to get another solution that does not contain bc. Then in this branch, we annotate all neighbors of a and d, delete $\{a, b, c, d\}$ from the graph, and include bc into the solution. Totally we can reduce p by at least $2 + 2q = 6$ by applying Lemma 11 to $abcd$ with $q = 2$. In the other two branches, we include $F = \{ab\}$ (resp., cd) into the solution, and can reduce p by at least $2 + 2q = 4$ by applying Lemma 11 to F with $q = 1$. We get

$$C(p) \leq C(p-6) + 2C(p-4), \tag{3}$$

which solves to $C(p) = O(1.2721^p)$.

If $r(b) + r(c) = 1$, say $r(b) = 1$ and $r(c) = 0$ without loss of generality, then we branch by including either ab or bc into the solution. Note that in the branch where bc is included into the solution, we can assume that $d \notin V(S^*)$ (for this case $r(d) = 0$), because if $d \in V(S^*)$ we can replace bc with ab in S^* to get another solution that does not contain bc. Then in this branch, we annotate all neighbors of d, delete $\{b, c, d\}$ from the graph, and include bc into the solution. Totally we can reduce p by at least 4 by applying Lemma 11 to bcd with $q = 1$. In the other branch where ab is included into the solution, we will delete all edges adjacent to ab from the graph and reduce p by at least 4 by applying Lemma 11 to ab with $q = 1$. Then we get the following recurrence

$$C(p) \leq 2C(p-4), \tag{4}$$

which solves to $C(p) = O(1.1893^p)$.

If $r(b) = r(c) = 1$, we branch by including bc into the solution or excluding it from the solution. In the first branch, we will delete $\{b, c\}$ from the graph, and reduce p by 2 by applying Lemma 11 to bc with $q = 0$. In the second branch, then ab and cd will be included into the solution. We will delete all edges incident on $\{a, b, c, d\}$ from the graph and reduce p by at least 6 by applying Lemma 11 to $abcd$ with $q = 2$. We get

$$C(p) \leq C(p-6) + C(p-2), \tag{5}$$

which solves to $C(p) = O(1.2107^p)$.

4.4 Branching on Maximal Inner Paths of Length ≥ 4 (Step 6)

Lemma 4 shows that there are only three kinds of maximal inner paths of length ≥ 4 after Step 2. For the three cases, we have a same branch rule. We use the same notation in Lemma 4 to denote the path. Let $abcde$ (or $abcdef$) be the path selected in Step 6, where $r(a) = r(b) = 0$ and $r(c) = 1$ hold in the three cases (i)-(iii) in Lemma 4. We branch into two branches according to the two cases: $a \notin V(S^*)$; and $a \in V(S^*)$. If $a \notin V(S^*)$, we can simply include bc into the solution. Then we annotate all neighbors of a, delete $\{a, b, c\}$ from the graph, and reduce p by 4 by applying Lemma 11 to abc with $q = 1$. Otherwise $a \in V(S^*)$ and we simply include cd into the solution. Then we can delete $\{c, d\}$ from the graph, annotate a, and reduce p by 2 by applying Lemma 11 to cd with $q = 0$. Therefore, we can branch on a maximal inner path with length ≥ 4 with recurrence (1).

4.5 Branching on Maximal Inner Paths of Length 2 (Step 7)

In Step 7, the algorithm will branch on a maximal inner path of length 2. If the graph has a maximal inner path abc of length 2 such that $r(b) = 1$, then our algorithm branches on it by including either ab or bc into the solution. We will delete $\{a, b\}$ (or $\{b, c\}$) from the graph. By applying Lemma 11 to ab (or bc) with $q = 1$, we can reduce at least 4 degree-3 vertices in each branch. We get a recurrence as (4).

Next, we assume that the graph only contains maximal inner paths abc of length 2 such that $r(b) = 0$. We call such maximal inner paths *bad 2-paths*. We distinguish the following three cases.

Case 1. There is a degree-3 vertex a such that only one bad 2-path is incident on a, say abb': The other two neighbors c and d of a (except b) are degree-3 vertices. We branch into four branches by either including each of ac, ad and ab into the solution or excluding all the three edges from the solution. When $F = \{ac\}$ (resp., ad) is deleted, we can reduce p by 6 by applying Lemma 11 to F with $q = 2$. For the branch of including ab into the solution, we can assume that $b' \notin V(S^*)$, otherwise we can replace ab with ac or ad to get another solution. Then we can also remove b' from the graph and annotate its neighbors. Totally, we will remove maximal inner path abb' from the graph and can reduce p by at least 6 by applying Lemma 11 on abb' with $q = 2$. For the branch of excluding all the three edges from the solution, we will include bb' into the solution. Then we will also remove abb' from the graph and can reduce p by at least 6 in this branch. We get the recurrence

$$C(p) \leq 4C(p-6), \tag{6}$$

which solves to $C(p) = O(1.2600^p)$.

Case 2. There is a degree-3 vertex a such that exactly two bad 2-paths are incident on a, say abb' and acc': The third neighbor d of a is a degree-3 vertex. For this case, we branch into four branches according to the four cases: $a \notin V(S^*)$; $b' \notin V(S^*)$; $c' \notin V(S^*)$; and $a, b', c' \in V(S^*)$. We look at the first branch ($a \notin V(S^*)$). We will delete a and annotate all neighbors of a. Then we will include bb' into the solution by applying reduction rules. Totally, we can remove inner path abb' and reduce p by at least 6 by applying Lemma 11 on abb' with $q = 2$. Analogously we can reduce p by at least 6 in the second and third branches. For the last branch, we simply include ad into the solution (note that $r(b) = r(c) = 0$ means that if ab or ac is in the solution with $a, b', c' \in V(S^*)$ then we can replace it with ad to get another solution) and annotate b' and c'. In this branch, we can reduce p by at least 6 by applying Lemma 11 on ad with $q = 2$. Therefore, we can get a recurrence as (6).

Case 3. There is no degree-3 vertex of Case 1 or Case 2: Now for every degree-3 vertex there are three bad 2-paths incident on it, and each edge in the graph is on a bad 2-path. We will arbitrarily select a bad 2-path abb' and branch on it into three branches according to the three cases: $a \notin V(S^*)$; $b' \notin V(S^*)$; and $a, b' \in V(S^*)$. Next, we analyze how many degree-3 vertices we can reduce in each branch. Assume that three bad 2-paths abb', acc' and add' are incident on a, where b', c' and d' are three different degree-3 vertices, since there is no good 1- or 2-edge cut. Let $N_2 = N_2(b') \cup N_2(c') \cup N_2(d') - \{a, b', c', d'\}$. If $|N_2| = 0$, then the component would contain only 4 degree-3 vertices. If $|N_2| = 1$, then G would have a good 1-edge cut. If $|N_2| = 2$, then G would have a good 2-edge cut. Therefore $|N_2| \geq 3$. We look at the branch corresponding to $a \notin V(S^*)$. In this branch, bb', cc' and dd' will be selected into the solution and deleted from the graph. In the remaining graph, all vertices in $N(b') \cup N(c') \cup N(d') - \{b, c, d\}$ become degree-1 vertices. After applying reduction rules, all degree-3 vertices in N_2 will be reduced. Totally, we can reduce at least 7 degree-3 vertices, which are $N_2 \cup \{a, b', c', d'\}$. By the same analysis, we know that in the branch corresponding to $b' \notin V(S^*)$, we can reduce p by at least 7. In the branch corresponding to $a, b' \in V(S^*)$, where $ab, bb' \notin S^*$ can be assumed, we delete b from the graph, annotate a and b', and reduce p by 2. Therefore, we get the recurrence

$$C(p) \leq 2C(p-7) + C(p-2), \tag{7}$$

which solves to $C(p) = O(1.2685^p)$.

4.6 Branching on 3-Regular Graphs (Step 8)

Assume that the graph is a 3-regular graph in Step 8, where the requirement of each vertex is either 1 or 0. We consider two cases: whether there is a vertex with requirement 1 or not. If all the vertices are of requirement 0, we arbitrarily select a vertex a (assume that b, c, d are the three neighbors of a), and branch into four branches by either including each of ab, ac and ad into the solution or excluding all the three edges from the solution. When $F = \{ab\}$ (resp., ac and ad) is included into the solution, we can reduce p by at least 6 by applying Lemma 11 to the inner path F with $q = 2$.

When none of the three edges is included into the solution, we can delete a from the graph and annotate $\{b, c, d\}$. By Lemma 9, we will reduce 4 degree-3 vertices. We consider the remaining graph G' after the fourth branch, which contains exactly three degree-2 vertices, each of which has requirement 1. If there is a good 1-edge cut in G', then we use Reduction Rule 5 to reduce it decreasing p by at least 2, and we will get a recurrence as (6). Next, we assume that G' contains no good 1-edge cut. We can find a degree-2 vertex v that is adjacent to two degree-3 vertices u and w (if the three degree-2 vertices are in a maximal inner path of length 4 in G', we apply Case 3 of Reduction Rule 4 to get a new degree-2 vertex v that is not adjacent to any other degree-2 vertices). Our algorithm will branch on uvw by including either uv or vw into the solution (not that $r(v) = 1$). In the branch of including uv in the solution, we remove $\{u, v\}$ from the graph, and we know that p is reduced by at least 4 just by removing the degree-3 vertex u by Lemma 9. In the other branch of including vw in the solution, we can also reduce p by at least 4. Then we can get the same recurrence (4). Combining all together, for this case we can get the following recurrence

$$C(p) \leq 3C(p-6) + 2C(p-8), \tag{8}$$

which solves to $C(p) = O(1.2721^p)$.

If the graph has a vertex a with $r(a) = 1$ (b, c, d being the three neighbors of a), we only need to branch into three branches by including each of ab, ac and ad into the solution and get a better recurrence

$$C(p) \leq 3C(p-6), \tag{9}$$

which solves to $C(p) = O(1.2010^p)$.

4.7 The Result

It is easy to see that (1) is one of the worst cases among all the cases, we get

Theorem 12: The annotated edge dominating set problem in graphs with maximum degree 3 can be solved in $O^*(1.2721^p)$ time and polynomial space, where $p \leq n$ is the number of degree-3 vertices in the graph.

5. The Parameterized Algorithm

We show that our exact algorithm presented in Sect. 4 can be used to get a fast algorithm for the *parameterized version* of the annotated edge dominating set problem, in which we are asked to decide whether an annotated edge dominating set instance in a degree-3 graph has a solution of size at most k or not. The following property for edge dominating sets is crucial for our algorithm.

Lemma 13: Let G be a graph with maximum degree 3. If G has p degree-3 vertices, then for any edge dominating set S, it holds $|S| \geq \frac{3}{10}p$.

Proof: Let $V(S)$ be the vertex set of an edge dominating set S, where $|V(S)| \leq 2|S|$ holds. Let p' be the number of degree-3 vertices in $V(S)$, and m' be the number of edges with one endpoint in $V(S)$ and the other in $V - V(S)$. We claim that $3(p-p') \leq m' \leq 3p' + 2(|V(S)| - p') - 2|S|$ holds. There are at least $p - p'$ degree-3 vertices in $V - V(S)$, and $V - V(S)$ is an independent set. Then we have $3(p-p') \leq m'$. In $V(S)$, there are $|V(S)| - p'$ vertices of degree ≤ 2. Then we have $m' \leq 3p' + 2(|V(S)| - p') - 2|S|$, as claimed. From the claim $3(p - p') \leq 3p' + 2(|V(S)| - p') - 2|S|$, we obtain $3p \leq 4p' + 2|V(S)| - 2|S| \leq 10|S|$, implying the lemma. □

Note that in Lemma 13, the edge dominating set is not required to be an annotated edge dominating set and the size of any annotated edge dominating set is not less then the size of a minimum edge dominate set. Based on Lemma 13, we can solve the parameterized version problem in degree-3 graphs in the following way: We first count the number p of degree-3 vertices in the input graph. If $k < \frac{3}{10}p$, we report that the graph has no (annotated) edge dominating set of size k. Otherwise we use our exact algorithm in Sect. 4 to find an minimum annotated edge dominating set in $O^*(1.2721^p) = O^*(1.2721^{\frac{10}{3}k}) = O^*(2.2306^k)$ time. Then we get

Theorem 14: There is an $O^*(2.2306^k)$-time and polynomial-space that decides whether a graph with maximum degree 3 has an annotated edge dominating set of size k or not.

6. Concluding Remarks

In this paper, we have presented an $O^*(1.2721^n)$-time algorithm for the annotated edge dominating set problem in degree-3 graphs, which is the fastest exact algorithm even for the edge dominating set problem in degree-3 graphs. Based on a kernelization result, the exact algorithm can also be used to get an $O^*(2.2306^k)$-time parameterized algorithm for the problem. Note that, recently, we have designed a parameterized algorithm for the edge dominating set problem in degree-3 graphs by using different techniques [25]. Together with our algorithms, we have presented some data reduction rules for the (annotated) edge dominating set problem, which can be used to reduce the input size of the graph. The annotated edge dominating set problem is a natural extension of the classical edge dominating set problem. It will be interesting to further study this problem in general graphs.

Many branch-and-search algorithms for graph problems have good performance when the graph has some high-degree vertices. So fast algorithms for the problems in low-degree graphs may directly lead to improvements on the algorithms for the problems in general graphs. This situation holds for independent set, vertex cover, edge dominating set and some other basic problems in graphs. It would be interesting to know whether there are faster algorithms for some basic graph problems when the graph is restricted to a low-degree graph.

Acknowledgments

This work was supported in part by the Scientific Grant-in-Aid from Ministry of Education, Science, Sports and Culture of Japan, and National Natural Science Foundation of China Grant No. 60903007.

References

[1] R. Beigel and D. Eppstein, "3-coloring in time $O(1.3289^n)$," J. Algorithms, vol.54, no.2, pp.168–204, 2005.
[2] D. Binkele-Raible and H. Fernau, "Enumerate and measure: improving parameter budget management," Proc. IPEC, LNCS 6478, Springer, pp.38–49, 2010.
[3] A. Bjorklund and T. Husfeldt, "Exact algorithms for exact satisfiability and number of perfect matchings," Algorithmica, vol.52, no.2, pp.226–249, 2008.
[4] N. Bourgeois, B. Escoffier, V.T. Paschos, and J.M.M. Rooij, "Maximum independent set in graphs of average degree at most three in $O(1.08537^n)$," TAMC LNCS 6108, Springer, pp.373–384, 2010.
[5] J. Chen, I.A. Kanj, and G. Xia, "Labeled search trees and amortized analysis: Improved upper bounds for NP-hard problems," Algorithmica, vol.43, no.4, pp.245–273, 2005.
[6] H. Fernau, "Edge dominating set: Efficient enumeration-based exact algorithms," H. Bodlaender, and M. Langston, eds., IWPEC, LNCS 4169, Springer, pp.142–153, 2006.
[7] F. Fomin, S. Gaspers, S. Saurabh, and A. Stepanov, "On two techniques of combining branching and treewidth," Algorithmica, vol.54, no.2, pp.181–207, 2009.
[8] F.V. Fomin, F. Grandoni, and D. Kratsch, "Measure and conquer: Domination - a case study," L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, eds., ICALP. LNCS 3580, Springer, pp.191–203, 2005.
[9] F.V. Fomin, F. Grandoni, and D. Kratsch, "Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm," SODA, pp.18–25, ACM Press, 2006.
[10] F.V. Fomin and K. Høie, "Pathwidth of cubic graphs and exact algorithms," Inf. Process. Lett., vol.97, no.5, pp.191–196, 2006.
[11] M.R. Garey and D.S. Johnson, "Computers and intractability: A guide to the theory of NP-completeness," Freeman, San Francisco, 1979.
[12] K. Iwama and T. Nakashima, "An improved exact algorithm for cubic graph tsp," G. Lin, ed., COCOON, LNCS 4598, Springer, pp.108–117, 2007.
[13] R. Niedermeier, "Invitation to Fixed-Parameter Algorithms," Oxford University Press, Oxford, 2006.
[14] V. Raman, S. Saurabh, and S. Sikdar, "Efficient exact algorithms through enumerating maximal independent sets and other techniques," Theory of Computing Systems, vol.42, no.3, pp.563–587, 2007.
[15] B. Randerath and I. Schiermeyer, "Exact algorithms for minimum dominating set," Technical Report zaik 2005-501, Universität zu Köln, Cologne, Germany, 2005.
[16] I. Razgon, "A faster solving of the maximum independent set problem for graphs with maximal degree 3," H. Broersma, S.S. Dantchev, and S. Szeider, eds., ACiD, vol.7, pp.131–142, Texts in Algorithmics., King's College, London, 2006.
[17] I. Razgon, "Exact computation of maximum induced forest," L. Arge and R. Freivalds, eds., SWAT. LNCS 5018, Springer, pp.160–171, 2006.
[18] I. Razgon, "Faster computation of maximum independent set and parameterized vertex cover for graphs with maximum degree 3," J. Discrete Algorithms, vol.7, no.2, pp.191–212, 2009.
[19] J.M. Rooij and H.L. Bodlaender, "Exact algorithms for edge domination," M. Grohe and R. Niedermeier, eds., IWPEC, LNCS 5018,

Springer, pp.214–225, 2008.
[20] R. Tarjan and A. Trojanowski, "Finding a maximum independent set," SIAM J. Comput., vol.6, no.3, pp.537–546, 1977.
[21] M. Xiao, "A simple and fast algorithm for maximum independent set in 3-degree graphs," Md.S. Rahman and S. Fujita, eds., WALCOM 2010, LNCS 5942, Springer, pp.281–292, 2010.
[22] M. Xiao, "A note on vertex cover in graphs with maximum degree 3," M.T. Thai, S. Sahni, eds., COCOON, LNCS 6196, Springer, pp.150–159, 2010.
[23] M. Xiao, "Exact and parameterized algorithms for edge dominating set in 3-degree graphs," W. Wu and O. Daescu, eds., COCOA (2), LNCS 6509, Springer, pp.387–400, 2010.
[24] M. Xiao and H. Nagamochi, "A Refined exact algorithm for edge dominating set," M. Agrawal, S.B. Cooper, and A. Li, eds., TAMC 2012, LNCS 7287, Springer, pp.360–372, 2012.
[25] M. Xiao and H. Nagamochi, "Parameterized edge dominating set in graphs with degree bounded by 3," Theor. Comput. Sci., 2012.
[26] M. Xiao, and H. Nagamochi, "Further improvement on maximum independent set in degree-4 graphs," W. Wang, X. Zhu, and D.-Z. Du, eds., COCOA, LNCS 6831, Springer, pp.163–178, 2011.
[27] M. Xiao, T. Kloks, and S-H. Poon, "New parameterized algorithms for the edge dominating set problem," Theor. Comput. Sci., 2012.
[28] M. Yannakakis and F. Gavril, "Edge dominating sets in graphs," SIAM J. Appl. Math., vol.38, no.3, pp.364–372, 1980.

Mingyu Xiao was born on December 30, 1979. He received his Ph.D. from the Chinese University of Hong Kong in 2008. He is an associate professor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include graph algorithms, optimization, parameterized complexity and so on.

Hiroshi Nagamochi was born in Tokyo, on January 1, 1960. He received the B.A., M.E. and D.E. degrees from Kyoto University, in 1983, in 1985 and in 1988, respectively. He is a Professor in the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University. His research interests include network flow problems and graph connectivity problems. Dr. Nagamochi is a member of the Operations Research Society of Japan, the Information Processing Society, and the Japan Society for Industrial and Applied Mathematics.